4对换、离散分配

2018年11月12日 8:30

-
- 对换Swapping

1. 最早用于麻省理工的单用户分时系统CTSS，又称交换技术

一. 多道程序环境下的对换技术

1. 对换的引入
   1) 阻塞、内存不足时外存程序不能进入内存，浪费系统资源，降低吞吐量
   2) 增加对换设施，把内存中暂不运行的进程换到外存，把外存其他进程换入
   3) 设置对换进程，把内存中暂时不运行的进程调进磁盘对换区；把磁盘上就绪进程调入，如Windows把程序在装入内存时若发现内存不足，也会自动对换
2. 对换的类型
   1) 整体对换/进程对换/整体对换：以进程为单位，即多道程序系统中级调度
   2) 页面对换/分段对换：以页面或分段为单位，目的是支持虚拟存储系统

二. 对换空间的管理

1. 对换空间管理的主要目标：长期存储的文件区、短暂驻留的对换区
   1) 文件区占磁盘大部分，访问频率低，主要目的提高空间利用率，离散分配
   2) 对换区占磁盘小部分，访问频率高，主要目的提高出入速度，连续分配
2. 对换区空闲盘块管理中的数据结构（类似内存动态分配）
   1) 空闲分区表，每个表目含首址和大小，由盘块号和盘块数表示
3. 对换空间的分配与回收：见内存动态分配与回收

三. 进程的换出与换入

1. 进程换出
   1) 选择阻塞、睡眠态进程，再选优先级最低的、有时还考虑内存中驻留时间
   2) 换出非共享程序、非共享数据段。若传送过程未出现错误，可回收其内存空间，修改对应数据结构，直到内存中再无阻塞进程运行
   3) 一般缺页且内存紧张时才启动对换，缺页率明显减少，吞吐量下降时停止
2. 进程换入
   1) 找出就绪态的已换出进程，再找换出时间最久的进程（一般规定大于2s以上的规定时间时才换入）若申请内存成功，直接调入，失败则继续换出
   2) 一般换入成功后会紧接着找其他换出进程，直到无足够内存换入新进程

-
- 分页存储管理方式

1. 分页存储：将逻辑空间分为若干固定大小区域，称为页/页面
   1) 典型的页面大小为1KB
   2) 相应地将内存空间分为若干物理块/页框frame，块和页大小相同
2. 分段存储：将逻辑空间分为若干大小不同的段，每段定义一组相对完整的信息
3. 段页式存储：上两种相结合，具有两者的优点

一. 分页存储管理的基本方法

1. 页面和物理块

1) 页编号从0开始，如第0页、第1页；块编号从0#开始，如第0#块、第1#块

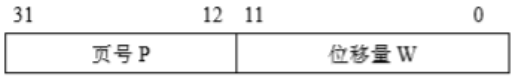
2) 进程分配时以块为单位，将若干页装入多个物理块

3) 最后一页常装不满一块，在块内形成“页内碎片”

4) 页面小，减少内存碎片空间，提高内存利用率；但页表过长，浪费内存，降低进程换进换出的效率。因而通常选1K~8K（通常是2的幂个b）

2. 地址结构

1)



2) 页号p，通常为12~31位，即一个地址空间有1M=2^20页

3) 位偏移量W，即页内地址，通常为0~11位

4) 逻辑地址空间中的地址为A，页面大小L，页内地址d，则：

(1)
$$P = INT\left[\frac{A}{L}\right]$$
$$d = [A] \ MOD \ L$$

5) 逻辑空间大小只受计算机地址位数限制

3. 页表/页面映像表：实现从页号到物理块号的地址映射

1)

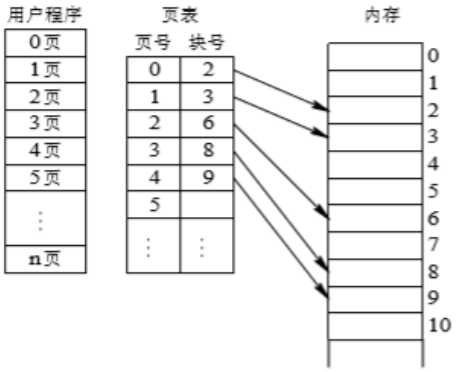


图 4-12 页表的作用

2) 存取控制字段：保护块中内容

(1) 一位字段规定内容可读写、只读

(2) 二位字段规定可读写、只读、只执行

(3) 进程试图写只读存储块时会引起操作系统一次中断

二. 地址变换机构

1) 实现从逻辑地址到物理地址的转换，由于页内地址和块内地址相同，实际上只需将逻辑地址空间页号转为内存物理块号。

2) 硬件参与的转换速度总是快于软件转换（类比交换机快于网桥）

1. 基本的地址变换机构

1) 在系统中设置页表寄存器(Page-Table Register)，存放页表始址，长度

2) 每条指令都需变换地址，但寄存器很贵，因而页表大多驻留在PCB

3) 调度进程时才更新页表寄存器，因而单处理机环境只需一个PTR

4) 变换步骤：

(1) 机构从有效地址/逻辑地址中分出页号，由硬件检索页表

(2) 若页号大于页表长度，说明越界，系统将产生地址越界中断

(3) 页表始址+页号*表项长度=该表项始址

(4) 把该表项的块号装入物理地址寄存器前几位

(5) 从有效地址/逻辑地址中分出页内地址，送入物理地址寄存器后几位

(6)

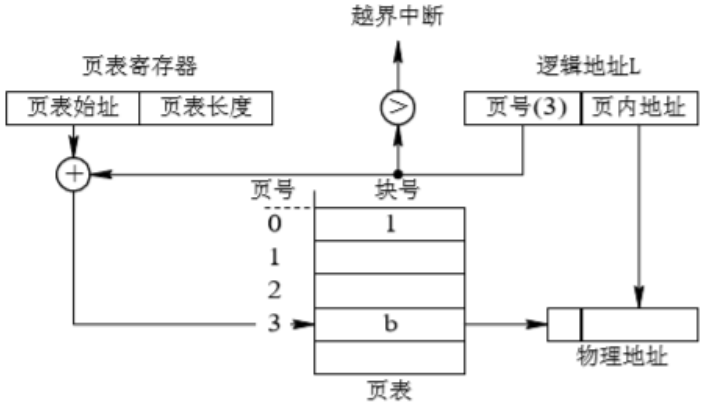


图 4-13　分页系统的地址变换机构

2. 具有快表的地址变换机构

1) 快表：为提高地址变换速度，在地址变换机构中增设一个具有并行查寻能力的特殊高速缓冲寄存器，又称为“联想寄存器”(Associative Memory)，还称Translation Lookaside Buffer

2) 新的步骤：

(1) CPU给出有效地址后，地址变换机构分出页号p，送入快表

(2) 检索出匹配页号后，直接把对应块号送入物理地址寄存器

(3) 未检索到，则重新访问内存中的页表，再加入快表

(4) 若快表已满，会挑出一个猜测不再需要的页表项，换出

(5)

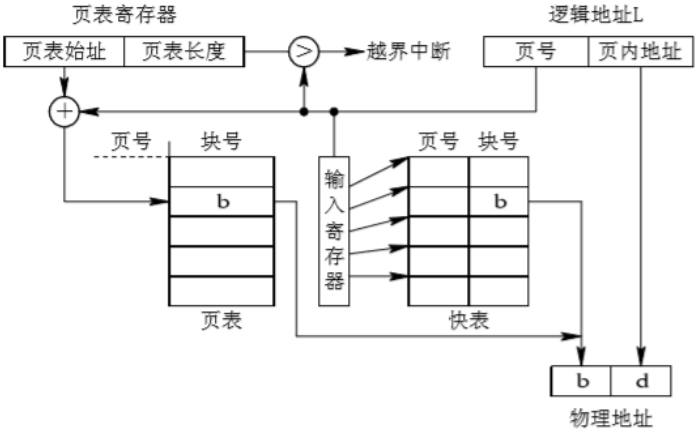


图 4-14　具有快表的地址变换机构

3) 通常快表也只放16~512个表项，但地址变换的速度损失已经可接受了

三. 访问内存的有效时间

1. 内存访问有效时间Effective Access Time：进程发出请求到操作数据所花时间
2. 无快表机构需要访问两次内存：先从页表算物理地址，再找数据
    1) 假设访问内存时间为t，则EAT=t+t=2t
3. 引入快表机构，需讨论在快表中与否，计算0-1分布数学期望
    1) 命中率a：在快表查到表项的比率
    2) 设查快表所需时间λ，则EAT=aλ+(1-a)(t+λ)=(2-a)t+aλ
    3) 例：设t=100ns，λ=20ns（纳秒，即一千分之一毫秒）

命中率a	0	50	80	90	98

有效访问时间EAT	220	170	140	130	122

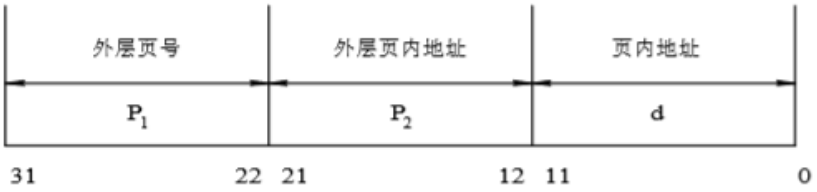
四. 两级和多级页表

1. 两级页表(Two-Level Page Table)

1) 现代计算机系统常支持逻辑地址空间2^32B~2^64B即2^2GB~2^34GB，页表将占用大量内存

2) 外层页表(Outer Page Table)：为离散分配的页表建立的页表

(1)



(2) 外层页表的每个页表项内存放的是页表分页的始址

(3) 最好再增设一个状态位S，若值为0，说明页表不在内存

3) 需增设外层页表寄存器存放外层页表的始址用外层页号做索引

4)

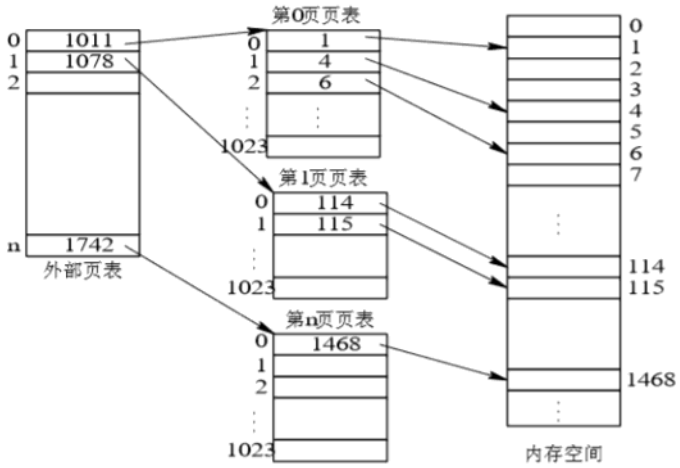


图 4-15 两级页表结构

5)

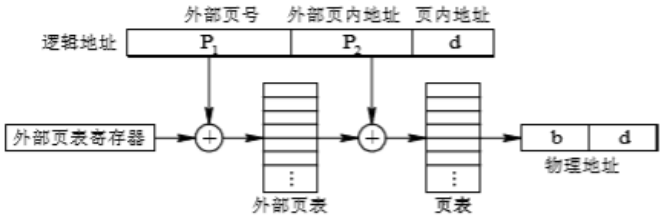


图 4-16 具有两级页表的地址变换机构

2. 多级页表

1) 64位机器的外层页号将有42位，表项将有4096G个，占空间16384G

2) 外层页表一般需要分页装入不同块，用第2级外层页表来映射外层页表

3) 64位机器支持2^64B=1844744TB规模的物理空间，也许需要很多级页表

4) 近年64位OS可直接把寻址的存储器空间减少为45位，三级页表就足够了

五. 反置页表Inverted Page Table

1. 反置页表的引入
   1) 为减少页表占用的内存空间，只为每个物理块设置一个表项，按块编号排序
      (1) 即用块编号索引，内容是页号及其隶属进程标识符
2. 地址变换
   1) 用进程标识符和页号检索反置页表，其序号i便是物理地址前几位
   2) 反置页表内容极少，未包含尚未调入内存的页面
   3) 每个进程需要一个外部页表External Page Table，调用不在内存的页面时会用到它
   4) 用进程标识符和页号检索线性表很费时，常利用Hash算法，但可能出现地址冲突

◆

◆ 分段存储管理方式

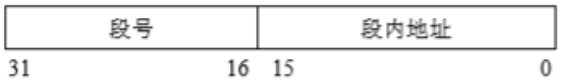
一. 分段存储管理方式的引入

1. 方便编程：按逻辑关系把作业划分成若干段
   1) 逻辑地址由段名/段号和段内偏移量/段内地址决定
      (1) LOAD 1，[A]|〈D〉将A段D单元内的值读入寄存器 1
      (2) STORE 1，[B]|〈C〉将寄存器 1 的内容存入B段C单元中
2. 信息共享：按逻辑给需要共享的一些信息建立段
3. 信息保护：同上，以逻辑单位信息为基础，标志读写属性
4. 动态增长：应对难以在运行前确定数据量的数据段
5. 动态链接：虚调用某段目标程序时，以段为单位调入目标程序并链接

二. 分段系统基本原理

1. 分段
   1) 通常用段号来代替段名，每段从0开始编址，采用一段连续的地址空间
   2) 段的长度取决于相应逻辑信息组的长度
   3) 每个段既包含地址空间，又标识了逻辑关系
   4) 逻辑地址由段号/段名和段内地址构成

(1)



2. 段表/段映射表
   1) 记录内存始址/基址和段的长度
   2) 实现从逻辑段到物理内存的映射

3)

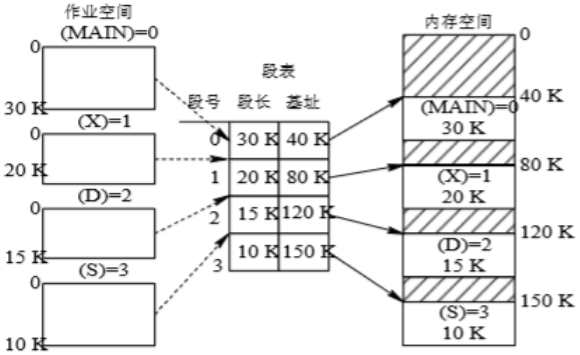


图 4-17 利用段表实现地址映射

3. 地址变换机构
   1) 段表寄存器：记录基址和表的长度TL
   2) 变换过程：
      (1) 若请求段号>TL，发出越界中断信号
      (2) 根据段表始址和请求段号，求出对应表项的位置，读出基址
      (3) 若段内地址d>段长SL，发出越界中断信号（图里漏了）
      (4) 段内地址d+=基址=需要的物理地址

(5)

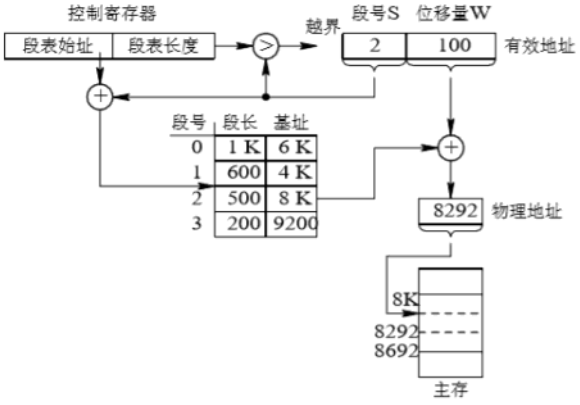


图 4-18　分段系统的地址变换过程

3) 也要访问两次内存，但段表项一般不多，所以没快表只会慢10%~15%

4. 分页vs分段：都是离散分配，都需要地址映射，但：
   1) 页是信息的物理单位，离散分布仅为提高内存利用率，是系统管理的需要
   2) 段是信息的逻辑单位，离散分布为保存一组完整信息，是用户的需要
   3) 页大小固定，由系统决定，由硬件结构划分的两部分实现
   4) 段大小不定，一般由编译程序根据信息的性质来划分
   5) 页的地址空间是一维线性的，只需一个记忆符即可表示地址
   6) 段的地址空间是**二维**的，标识地址既需段名，又需段内地址

三. 信息共享

1. 分页系统中程序和数据的共享
   1) **可重入Reentrant：能被共享**（无论在分页还是分段系统中）
      (1) 如TextEditor的，占了两个8m的ed1，ed2页和一个8m的data1页

(2)

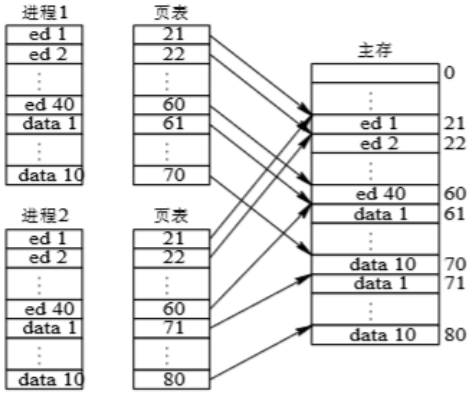


图 4-19　分页系统中共享 editor 的示意图

2. 分段系统中程序和数据的共享

1)

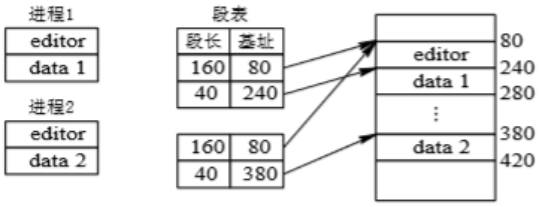


图 4-20　分段系统中共享 editor 的示意图

   2) 可重入代码Reentrant code/纯代码Pure code：能被多进程同时访问的代码
   3) 纯代码不允许在执行中做出任何改变
      (1) 若每个进程都有局部数据区，用于存储共享数据代码数据的副本，就能保证共享代码成为可重入代码

四. 段页式存储管理方式

1. 基本原理：

1) 将用户程序先分成若干个段并赋予段名，各段再分成若干个页

(1)

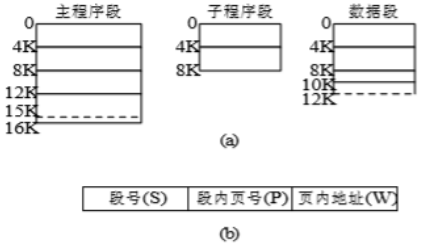


图 4-21 作业地址空间和地址结构

2) 段表项内容从内存始址和段长改为页表始址和页表长

(1)

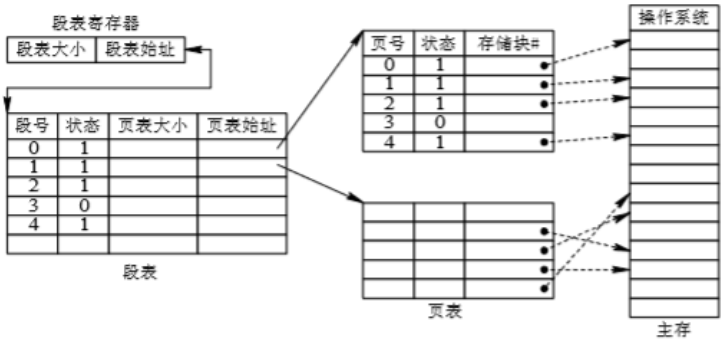


图 4-22 利用段表和页表实现地址映射

2. 地址变换过程

1) 如图

(1)

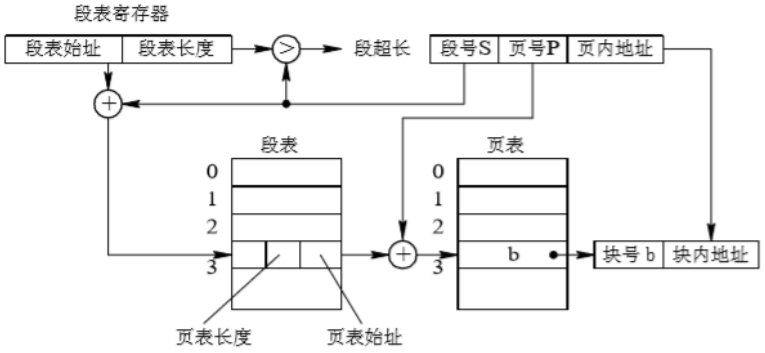


图 4-23 段页式系统中的地址变换机构

2) 访问指令/数据需要**三次访问内存：段表找页表，页表找块号，正式访问**

(1) 因而要用快表，同时利用段号和页号检索高速缓存

i.

ii.

iii.

iv.

v.

vi.

vii.

viii.

ix.

x. ---------我是底线-------